Add networkx to REQUIREMENTS, compile, and then execute the next block of code exactly as written:

```
********************************************************************************
# Input (Karate Club): Load and visualize Zachary's Karate Club graph
# Step-by-step, well-documented, clear output for downstream use
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from collections import Counter

random_seed = 1234

# 1. Load the Karate Club graph
GRAPH = nx.karate_club_graph()

GRAPH = nx.Graph(GRAPH)

# 2. Assign 'ground_truth_string' attribute to each node, based on 'club' (either 'Mr. Hi' or 'Officer')
for node in GRAPH.nodes:
# The 'club' attribute is present in each node's dictionary
GRAPH.nodes[node]['ground_truth_string'] = GRAPH.nodes[node]['club']

# 3. Color nodes by club membership
club_names = list(sorted({GRAPH.nodes[n]['club'] for n in GRAPH.nodes}))
club_colors = {'Mr. Hi': '#1f77b4', 'Officer': '#ff7f0e'}
node_colors = [club_colors[GRAPH.nodes[n]['ground_truth_string']] for n in GRAPH.nodes]

# 4. Plotting with spring layout
plt.figure(figsize=(8,6))
pos = nx.spring_layout(GRAPH, seed=42)
nx.draw(
```

```
GRAPH, pos,
node_color=node_colors,
with_labels=True, font_size=8,
node_size=400, edge_color='#cccccc', linewidths=1.5
)
# Legend mapping color to club
handles = [mpatches.Patch(color=club_colors[club], label=club) for club in club_names]
plt.legend(handles=handles, title="Club (Ground Truth)", loc='upper right')
plt.title("Zachary's Karate Club: Nodes colored by 'club' membership")
plt.axis('off')
plt.tight_layout()
plt.show()

# 5. Print out summary of nodes in each club
club_counter = Counter([GRAPH.nodes[n]['ground_truth_string'] for n in GRAPH.nodes])
print('Node count by club:')
for club, count in club_counter.items():
print(f" {club}: {count}")

# 6. Show a sample of club assignments
print("\nSample ground_truth_string assignments (node: club):")
for n in range(6):
print(f" Node {n}: {GRAPH.nodes[n]['ground_truth_string']}")

# Export main outputs for downstream use
graph_karate = GRAPH
karate_club_counter = club_counter
karate_club_names = club_names
karate_club_colors = club_colors
karate_layout = pos


**********************************************************************************
```

Prompt starts here:

I have organized my instructions into sections.
Each section should be contained in a single block.
Each block should be named according to what follows the "Start Block" text.
Use the imported network data in the GRAPH object.

Start Block "Define Objective Functions"
****
Define two objective functions:
#1. Modularity, calculated using the networkx modularity function.
#2. Log-likelihood, defined below:

Start Log-Likelihood Algorithm Definition

Log-Likelihood Part 1. Assume the number edges between nodes within each of the K groups is distributed according to
a binomial distribution, with parameters:
number of trials = N_i choose 2
success probability = (O_i)/ (N_i choose 2)
, where
N_i = the number of nodes within the i^th group
O_i = the number of observed, undirected edges between the N nodes of the group

For each of the K groups, calculate the probabiliity P_i of observing O_i edges within the i^th group.

Log-Likelihood Part 2. Assume the number edges between nodes between all of the K groups is distributed according to
a binomial distribution, with parameters:
number of trials = N choose 2
success probability = (O_bw)/ (N choose 2)
, where
N = the number of nodes in the network.
O_bw = (the number of observed, undirected edges between all nodes in the network) - (the sum of the O_i from "Log-Likelihood Part 1" for all K

groups)

Calculate the probability P_bw of observing O_w edges between all of the k groups.

Log-Likelihood Part 3. Log-likelihood is calculated as the sum of the natural log of P_i, for all i in K, and the natural log of P_bw.
Likelihood is calculated as the product of P_i, for all i in K, and P_bw.

End Log-Likelihood Algorithm Definition
****
End Block "Define Objective Functions"
All of the above must be executed in a single block.

Start Block "Ground Truth Objective Values"
****
Using the previously-created objective functions:
1) Calculate the modularity of the ground-truth group assignment.
2) Calculate the log-likelihood of the ground-truth group assignment.
****
End Block "Ground Truth Objective Values"
All of the above must be executed in a single block.

Start Block "Evaluate Multiple K values"
****
Use the previously-defined random seed.
Letting starting_solution be a random assignment of the N nodes into K groups.

When performing simulated annealing, use the following cooling schedule:
Number of iterations=20,000
Initial temperature=5
Alpha=0.995
Minimum temperature=1e-5

For each value of K=2 to 10:
{

Using simulated annealing, identify the approximately optimal assignment of the N nodes into K groups that maximizes modularity.
The simulated annealing process should begin at the starting_solution group assignment.

Require that all groups in the solution have atleast 2 nodes.
Save the modularity of each optimal solution, along with the associated K value.
Save each optimal group assignments, along with the associated K value.
All variables created during the modularity maximization process should have suffix "_mod" added to their name.
}

For each value of K=2 to 10:
{

Using simulated annealing, identify the approximately optimal group assignment of the N nodes into K groups that maximizes log-likelihood.
The simulated annealing process should begin at the starting_solution group assignment.

Require that all groups in the solution have atleast 2 nodes.
Calculate the Bayesian Information Criterion of the final solution.
Save the log-likelihood of each optimal solution, along with the associated K value.
Save the Bayesian Information Criterion of each optimal solution, along with the associated K value.
Save each optimal group assignments, along with the associated K value.
All variables created during the log-likelihood maximization process should have suffix "_lik" added to their name.
}

Use the same cooling schedule for both modularity and log-likelihood annealing.
The code for both modularity and log-likelihood annealing must be in a single block.
****
End Block "Evaluate Multiple K values"
All of the above must be executed in a single block.

Start Block "Plot Results"
****
Create a single figure containing two subplots, vertically-aligned, one for modularity and one for log-likelihood.

For modularity:
Print a single line plot showing the modularity of each optimal solution as a function of K.
Select the final modularity solution to be the solution that had the highest modularity.

For log-likelihood:
Print a single line plot of the Bayesian Information Criterion of each optimal solution as a function of K.
Select the final log-likelihood solution to be the solution that had the lowest Bayesian Information Criterion.
****
End Block "Plot Results"
All of the above must be executed in a single block.

Start Block "Plot Optimized Network Groupings"
****
Create a single figure containing three subplots, vertically-aligned:
1. A plot of the network using the ground-truth group assignment.
Use the pos paramter defined above in the networkx draw function.
Color each node according to its group assignment. The legend should relate the group color to the group name as defined in ground_truth_string.
Each group should have a distinct color assignment.
Provide a suitable title for the plot.

2. A plot of the network using the modularity-maximizing group assignment.
Color each node according to its group assignment.
Use the pos paramter defined above in the networkx draw function. All nodes should be in the same position as above.
Use the same color scheme for groups as above. The legend should relate the group color to the group name as defined in ground_truth_string.
Provide a suitable title for the plot.

3. A plot of the network using the log-likelihood-maximizing group assignment.

Color each node according to its group assignment.
Use the pos paramter defined above in the networkx draw function. All nodes should be in the same position as above.
Use the same color scheme for groups as above. The legend should relate the group color to the group name as defined in ground_truth_string.
Provide a suitable title for the plot.
****
End Block "Plot Optimized Network Groupings"
All of the above must be executed in a single block.